PAPER REVIEW

How to Compute Fast a Function and All Its Derivatives

A variation on the theorem of Bauer-Strassen

Jean-Baptiste KECK

M2 MSIAM

February 1, 2015

1 Introduction

This paper is about an alternative algorithmic proof of the Baur-Strassen result given and proved in [**2**], showing that the complexity of the evaluation of a rational function of several variables and all its derivatives is bounded above by three times the complexity of the evaluation of the function itself. The new proof is constructive and much simpler than the original one in [**2**].

2 Notations

The notation is mainly the one used in the original paper [1] and [2] with some minor changes.

$\mathbb{K}$ is an infinite field.
$F \in \mathbb{K}_n(X)$ is a rational function of n variables $x_1, x_2, ..., x_n$.
$\widetilde{F} \in \mathbb{K}_{n+1}(X)$ is a rational function of $n+1$ variables $x_1, x_2, ..., x_n, y$.

The set of partial derivatives of F is denoted $F' = \left\{ \frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, ..., \frac{\partial F}{\partial x_n} \right\}$.

Let $\odot$ be an arithmetical operation and $x = (x_1, x_2, ..., x_n) \in \mathbb{K}^n$.

$$(\text{a} \odot \text{b is an } \textbf{essential operation}) \Leftrightarrow \begin{cases} a = f(x) \text{ and } b = g(x) \text{ with } f, g \in \mathbb{K}_n(X) \\ \textbf{or } a \in \mathbb{K}, b = g(x) \text{ and } \odot \text{ is a division.} \end{cases}$$

Let $\mathcal{A}$ be an algorithm computing F from $x = (x_1, x_2, ..., x_n)$ and $\mathbb{K}$. We have :

$$\exists\ u \in \mathbb{N}\ \ \mathcal{A} = \{g_k, g_2, ..., g_u\} \text{ with } \begin{cases} g_k \in \{x_1, x_2, ..., x_n\} \cup \mathbb{K} \\ \textbf{or } g_k = g_{k_1} \odot g_{k_2} \text{ with } k_1, k_2 < k \end{cases} \tag{1}$$

Finally let :

- $s(F)$ be the number of **essential multiplications and divisions** in $\mathcal{A}$.
- $m(F)$ be the total number of **multiplications and divisions** in $\mathcal{A}$.
- $T(F)$ the total number of **essential operations** in $\mathcal{A}$.
- $\Theta(F)$ the total number of **operations** in $\mathcal{A}$.

3 Baur-Strassen's Theorem

Theorem: From each algorithm $\mathcal{A}$ computing F one can derive and algorithm $\mathring{\mathcal{A}}$ computing F and F' such that :

$$(P) \Leftrightarrow \begin{cases} s(F, F') & \leq & \mathbf{3}\, s(F) \\ m(F, F') & \leq & \mathbf{3}\, m(F) \\ T(F, F') & \leq & \mathbf{5}\, T(F) \\ \Theta(F, F') & \leq & \mathbf{5}\, \Theta(F) \end{cases} \tag{2}$$

Those inequalities are independent of the number of variables n.

4 Proof overview

Foreword: *Many errors have been corrected from the original paper and I tried to make some parts clearer so that it appears less confusing. However, new errors might as well have been introduced too.*

The proof is made by **induction on the length of the algorithm**. Let $\mathcal{A}_u$ be an algorithm of length u computing a rational function F and g_k be the result of the first operation of $\mathcal{A}_u$.

Define $\widetilde{F}$ a function of $n+1$ variables such that $F(x_1, x_2, ..., x_n) = \widetilde{F}(x_1, x_2, ..., x_n, \mathbf{y})$ with $y = g_k(x_1, x_2, ..., x_n)$.

$\mathcal{A}_u$ induces an algorithm $\mathcal{A}_{u-1}$ which computes $\widetilde{F}$ from $x_1, x_2, ..., x_n, g_k$ and $\mathbb{K}$ in one less operation ($\mathcal{A}_{u-1}$ is of length $u-1$). **By induction hypothesis, $\widetilde{F}$ satisfies (P).**

We have $\forall h \in [\![1, n]\!]$ $\dfrac{\partial F}{\partial x_h} = \dfrac{\partial \widetilde{F}}{\partial x_h} + \dfrac{\partial \widetilde{F}}{\partial y} \cdot \dfrac{\partial g_k}{\partial x_h}$ (1).

The idea is then to examine all six possible cases :

1. Example case $\mathbf{g_k} = \mathbf{c} \cdot \mathbf{x_i}$ where $c \in \mathbb{K}$ and $i \in [\![1, n]\!]$:

$$(1) \Rightarrow \begin{cases} \dfrac{\partial F}{\partial x_h}(x_1, ..., x_n) = \dfrac{\partial \widetilde{F}}{\partial x_h}(x_1, ..., x_n) + c \cdot \dfrac{\partial \widetilde{F}}{\partial y}(x_1, ..., x_n) & \text{if } h = i \\[2ex] \dfrac{\partial F}{\partial x_h}(x_1, ..., x_n) = \dfrac{\partial \widetilde{F}}{\partial x_h}(x_1, ..., x_n) & \text{if } h \neq i \end{cases}$$

$$\text{In this case we have :} \begin{cases} s(F, F') & = & s(\widetilde{F}, \widetilde{F}') \\ m(F, F') & \leq & m(\widetilde{F}, \widetilde{F}') + 1 + 1 \\ T(F, F') & \leq & T(\widetilde{F}, \widetilde{F}') + 1 \\ \Theta(F, F') & \leq & \Theta(\widetilde{F}, \widetilde{F}') + 1 + 1 + 1 \end{cases}$$

2. Case $\mathbf{g_k} = \mathbf{x_i} \cdot \mathbf{x_j}$ where $i, j \in [\![1, n]\!]$.

$$(1) \Rightarrow \begin{cases} \dfrac{\partial F}{\partial x_h}(x_1, ..., x_n) = \dfrac{\partial \widetilde{F}}{\partial x_h}(x_1, ..., x_n) + x_j \cdot \dfrac{\partial \widetilde{F}}{\partial y}(x_1, ..., x_n) & \text{if } h = i \\[2ex] \dfrac{\partial F}{\partial x_h}(x_1, ..., x_n) = \dfrac{\partial \widetilde{F}}{\partial x_h}(x_1, ..., x_n) + x_i \cdot \dfrac{\partial \widetilde{F}}{\partial y}(x_1, ..., x_n) & \text{if } h = j \\[2ex] \dfrac{\partial F}{\partial x_h}(x_1, ..., x_n) = \dfrac{\partial \widetilde{F}}{\partial x_h}(x_1, ..., x_n) & \text{if } h \notin \{i, j\} \end{cases}$$

$$\text{In this case we have :} \begin{cases} s(F, F') & \leq & s(\widetilde{F}, \widetilde{F}') + 2 + 1 \\ m(F, F') & \leq & m(\widetilde{F}, \widetilde{F}') + 2 + 1 \\ T(F, F') & \leq & T(\widetilde{F}, \widetilde{F}') + 2 + 2 + 1 \\ \Theta(F, F') & \leq & \Theta(\widetilde{F}, \widetilde{F}') + 2 + 2 + 1 \end{cases}$$

We do exactly the same for the 4 other cases :

3. Case $\mathbf{g_k} = \mathbf{c}/\mathbf{x_i}$ where $c \in \mathbb{K}$ and $i \in [\![1, n]\!]$.
4. Case $\mathbf{g_k} = \mathbf{x_i}/\mathbf{x_j}$ where $i, j \in [\![1, n]\!]$.
5. Case $\mathbf{g_k} = \mathbf{x_i} + \mathbf{d}$ where $d \in \mathbb{K}$ and $i \in [\![1, n]\!]$.
6. Case $\mathbf{g_k} = \mathbf{x_i} + \mathbf{x_j}$ where $i, j \in [\![1, n]\!]$.

Putting everything together and using (P) for $\widetilde{F}$, we get at most the inequalities:

$$\begin{cases} s(F,F') & \leq & s(\widetilde{F},\widetilde{F}') & + & 3 & \leq & 3\,s(\widetilde{F}) & + & 3 & = & 3\,s(F) \\ m(F,F') & \leq & m(\widetilde{F},\widetilde{F}') & + & 3 & \leq & 3\,m(\widetilde{F}) & + & 3 & = & 3\,m(F) \\ T(F,F') & \leq & T(\widetilde{F},\widetilde{F}') & + & 5 & \leq & 5\,T(\widetilde{F}) & + & 5 & = & 5\,T(F) \\ \Theta(F,F') & \leq & \Theta(\widetilde{F},\widetilde{F}') & + & 5 & \leq & 5\,\Theta(\widetilde{F}) & + & 5 & = & 5\,\Theta(F) \end{cases}$$

$\Rightarrow$ (P) for F, which proves the theorem since the beginning of the induction is trivial.

Those inequalities are independent of the number of variables n since we just added a bounded number of new operations.

5 Conclusion

From this new proof of the theorem we can deduce a constructive recursive backward way to build an algorithm for the computation of all partial derivatives of F at a given point x from any algorithm computing F at this point.

This paper might be the starting point for autodifferentation techniques used to numerically evaluate the derivative of a function specified by a computer program. Autodifferentiation exploits the fact that every computer program, no matter how complicated, executes a sequence of elementary arithmetic operations (addition, subtraction, multiplication, division) like in this paper but with additional elementary functions (exp, log, sin, cos, etc.). By applying exactly the same chain rule repeatedly to these operations, derivatives of arbitrary order can be computed automatically, using at most a small constant factor more arithmetic operations than the original program.

In fact I found out that Jacques Morgenstern, the author of this paper, was initiator of the SAPHIR (Systèmes Algébriques Formels pour l'Industrie et la Recherche) research project with André Galligo. This project leaded to the creation of autodifferentiation tool Odyssée which targeted `Fortran` industrial code. Today this tool has been renamed to Tapenade and is developed and maintained by Inria. It can generate forward (tangent) or reverse (adjoint) differentiated programs for `C`, `C++` or `Fortran`.

References

[1] Jacques Moergenstern. How to compute fast a function and all its derivatives. *ACM SIGACT News*, 16:60–62, 1985.

[2] Walter Baur; Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.